Casper: extraction of a ledger from a tree of blocks
26/3/2021

1 "Proof of stake": a variation on permissioned systems

Let us remind that, in PBFT, the replicas are numbered from 1 to n and are fixed since the beginning of the protocol. No other replica is allowed to take part to the protocol. This constraint is denoted as a "permissioned system". Consider now the following variation of PBFT. Consider that there exists a good, denoted as the "voting power". One unit of voting power is denoted as a "voting token" (vt). We assume that the total quantity of voting power in the world is fixed and equal to N vt. Replicas can own vt and transfer them to each other. We assume that there exists a public register that makes public which quantity of vt belongs to which replica in the world. For simplicity wa assume that this register is fixed from the beginning until the end of the protocol.

Consider that, now, in PBFT there is not anymore a predetermined number n of replicas. Now, it is not required anymore to collect a fixed number of messages, e.g. $> 2n/3$, to perform some action (e.g. send a "commit" message). Instead, the parameter is to collect a number of messages, such that, the total money owned by the replicas who sent them, is $> 2/3N$. With this new protocol, the safety and liveness properties of PBFT are now guaranteed as long as $> 2/3N$ of the money in the world is owned by honest replicas.

Let us make some examples of voting power in real-life. First, in Ethereum, the public register is implemented by writing in blocks the quantity of money, denoted as "Ether" owned by each client (a client is denoted as a "wallet"). In the Casper protocol for Ethereum, that we are going to see, the replicas, which are called "validators", are publicly given voting power as follows. They initially pledge, before they join the Casper protocol, to some amount of the Ether that they own in their wallet. Proportionnally to this money pledged, they are given voting power in the protocol as above. If they behave honestly, then they are rewarded by some money at the end of the protocol. If they behave badly, then the amount of money to which they pledged is deleted from their account, which is denoted as "slashed". In the Tezos protocol, there is a public process denoted as a "lottery", which automatically gives vt to some owners of a certain cryptocurrency denoted as "Tezos". The chances to win vt at this lottery are proportionnal to the amount of Tezos owned. vt are then transferable. For instance, from a replica with many vt who does not want to participate, into a smaller replica who is willing to participate. This is denoted as "delegation of voting power".

2 Casper: PBFT adapted to trees of blocks

Casper [BG19] is a layer over any mechanism that produces a tree of blocks (such as Bitcoin). Its goal is to provide a read function, alternative to the one of Bitcoin, that returns a ledger. The safety purpose of Casper is the same as in Bitcoin. Namely, prevent that two conflicting branches may be read in the same execution. To achieve this, Casper increasingly "finalizes" a branch: the branch up to the highest "finalized" block will be by definition read as the ledger. In a nutshell, the advantage of Casper is that it removes the need of a mining function, such as in Bitcoin where honest nodes must constantly mine more than dishonest nodes to maintain safety. On the other hand, Casper requires a permissioned setting, for instance, in the previous sense of "proof of stake". Also, liveness of Casper requires some specific honest behavior of the mechanism that creates blocks, coupled with timing assumptions.

2.1 Model

2.1.1 Validators in an asynchronous network

We consider processes denoted as "validators". There exists a public register that assigns, to each validator, a certain amount of voting tokens (vt). The total number of vt is denoted N. The validators are in an asynchronous network. That is, they can send authenticated messages to each other, but there is no guarantee on the time taken to deliver a message.

2.1.2 Data structure: tree of blocks, links, vote messages

Reminder of Bitcoin: blocks organized in a tree We consider the same data structure as in Bitcoin. Recall that the basic data structure is denoted as a "block", which itself contains strictly ordered values. Each block has a father, and all blocks have the same common ancestor, denoted as the "genesis block". In figure 1, the genesis block is denoted is r (for "root"). The black arrows in figure 1 are unfortunately in the wrong sense, since they point from the child to the father. The length of the path from the genesis block to a block b is denoted is the *height* of b, and denoted as $h(b)$. In particular, all blocks belong to the same tree structure. Recall that two blocks are denoted as *conflicting* if they are not on the same branch. Said otherwise, neither is a descendent nor an ancestor of the other.

Links A link is the data of an ordered pair of blocks: (s, t) such that s is an ancestor of t. s is denoted the "source" and t the "target". On figure 1, the pink arrows are examples of links, which point from the source to the target.

vote messages Validators can issue special messages of type denoted vote, that contain a *link* as argument (and not a block!).

A supermajority link Is a link for which there exists vote messages issued by validators owning more than $2/3N$ vt. Said otherwise, which is voted by more than $2/3$ of validators weighted by their voting power.

A block b is denoted as *justified* If it satisfies the following recursive definition: 1) either it is the genesis block, or 2) if there exists a supermajority link from source a justified block and target b. For instance on figure 1, the supermajority links are denoted in pink. In particular, there is a consecutive sequence of supermajority links from the genesis block: r, b_1, b_2 and b_3. Thus by definition, all blocks r, b_1, b_2 and b_3 are justified.

A block b is denoted as *finalized* If 1) either it is the genesis block 2) or we have simultaneously that b is justified, and is itself the source of a supermajority link with target at height $h(b) + 1$. Said otherwise, of a supermajority link of length 1. In particular, on figure 1, none of the justified blocks is the source of a supermajority link of length 1. So no block is finalized, excepted the genesis block.

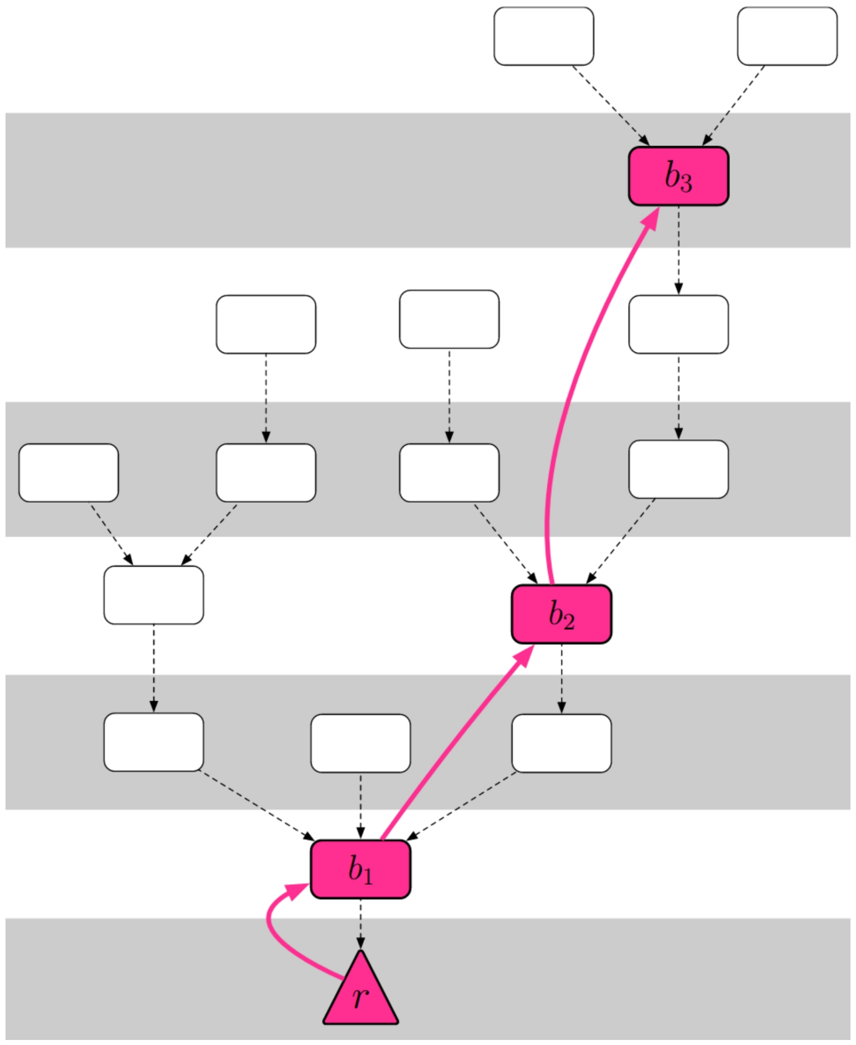

Figure 1: r, b_1, b_2 and b_3 are justified. No block is finalized

2.2 Casper protocol (Figure 2)

Notice that we differ from the presentation of [BG19], in that we also incorporate in the protocol the fork choice rule. Indeed, it turns out that this rule is *necessary* to prove Theorem 2 below.

Also, anticipating on the proof of Corollary 3, notice in particular that the "Liveness rule" for the block production mechanism guarantees that, if the block production mech-

Casper protocol

We consider a fixed duration Δ, which is a parameter of the protocol.

Read To read the ledger, a process *queries* all the validators to send to it their respective trees of blocks. Then it merges them, then extracts from it the sub-branch: from the genesis block, until the finalized block of greatest height. Unicity of this branch is guaranteed by Theorem 1. Then, by definition, the read operation returns the ordered sequence of values in the blocks of this sub-branch.

"Safety rules" for validators validators must *not* send a vote message for two links (s_1, t_1) and (s_2, t_2) such that either

I) t_1 and t_2 are conflicting, and at the same height: $h(t_1) = h(t_2)$;

II) or, $h(s_1) < h(s_2) < h(t_2) < h(t_1)$. Said otherwise, it must not vote for heights which are strictly within the span of another vote of him.

"Liveness rule" for validators, a.k.a. the "fork choice rule" Validators must do *exactly* the following, and *nothing* else:

a) Consider the branch of their local tree, that contains *the* justified block a of greatest height. [Unicity of this branch & block is guaranteed if $> 2/3$ of validators, weighted by their vt, are honest, since they then respect the safety rule I)].

b) Create vote messages for every possible link with *source* equal to this block a, until they cannot create anymore votes. [Either: because there is no more link to vote for, or: because this would violate the Safety rules, given the votes that he already created.]

c) Send all these votes to all validators.

"Liveness rule" for the block production mechanism Request all the local trees of validators and merge them. Consider *the* justified block a of greatest height, as considered just above. Then, consider the greatest height m such that there exists a block b_m which is the target of at least one vote. Create *exactly* one new branch of blocks: from a, until a block a_{m+1} at height $m + 1$. Then *do nothing* during a certain delay Δ.

Figure 2: Casper protocol

anism follows it forever, then, there will not exist any other descendent a'_{m+1} of same height $h(a'_{m+1}) = h(a_{m+1}) = m+1$, that would conflict with a_{m+1}.

2.3 Safety of Casper

Theorem 1. *Assume that strictly more than 2/3 of validators, weighted by their voting power (vt), are honest, in particular that they respect the safety rules. Then, no two conflicting blocks can ever be finalized. In particular, no two conflicting ledgers can be read.*

The idea of the proof, which is done in [BG19] bottom of page 4, is that, since a finalized block a_3 must have a father which is justified, then, if a conflicting finalized block b_3 exists, then, we must be in the situation of Figure 3. In particular, there would exist a supermajority link (a_2, a_3) with heights strictly within the span of another supermajority link (b_2, b_3). But this is impossible, since for this to happen this would mean that at least one honest validators violated the Safety rule II.

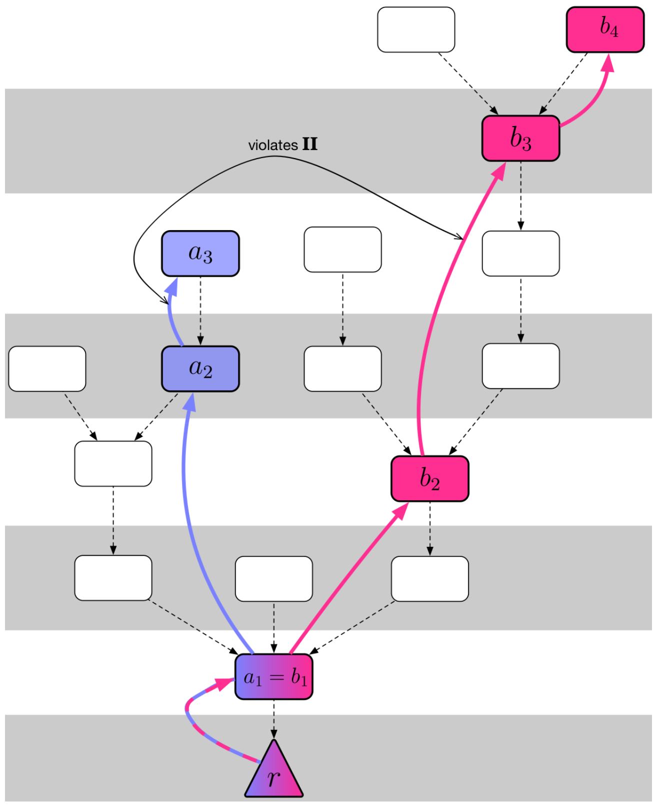

Figure 3: Proof of Thm 1: assuming by contradiction existence of two conflicting finalized blocks a_3 and b_3.

2.4 Liveness of Casper, under partial synchrony + honest blocks mechanism

Liveness is not guaranteed by just assuming that $> 2/3$ of the validators, weighted by vt, are honest. Still, under just this assumption, then we have at least the following *possibility* result. Notice that this result is an implicit ingredient in the proof of [BG19, Theorem 2].

Theorem 2. *Consider the* justified *block a of greatest height. Then, consider the greatest height m such that there exists a block b_m which is the target of at least one vote. Consider a descendent a'_{m+1} of a of height $m+1$. Then, all honest validators* can *vote for the link (a, a_{m+1}) without violating the Safety rules.*

Proof. First, voting (a, a_{m+1}) does not violate I), since no (honest) validator voted for a link with target at height $m+1$. Second, no honest validator voted for a link with heights strictly within the span $[h(a), h(a_{m+1}) = m+1]$. Indeed, by the liveness rule, since a is the highest justified block, then, validators cannot vote for a link with source of strictly higher height than a. □

Corollary 3. *Suppose in addition that, from some point in time, the block production mechanism becomes honest,* and*, that validators have the time to diffuse to all the network all the votes that they are required to send, within delay Δ. Then Casper is live.*

Proof. A block production mechanism following the Liveness rule will always create exactly the block a_{m+1} required by the previous theorem, then wait Δ. During this delay Δ, validators will be able to make a_{m+1} justified. Indeed, the previous Theorem 2 guarantees that they *can* vote for the link (a, a_{m+1}). And, by honesty of the block production mechanism, no conflicting block a'_{m+1} exists. Thus, all the votes of honest validators will necessarily be concentrated on the link (a, a_{m+1}).

Then, after Δ is elapsed, the block production mechanism will create a direct child of a_{m+1}, which will also become justified, thus making a_{m+1} a finalized descendent of a. □

References

[BG19] Vitalik Buterin and Virgil Griffith. *Casper the Friendly Finality Gadget.* 2019. arXiv: 1710.09437 [cs.CR].